Subject: Re: RPC_METHOD how to "define"
Posted by dolik.rce on Sun, 05 Feb 2017 19:48:32 GMT
View Forum Message <> Reply to Message

NilaT wrote on Sun, 05 February 2017 11:02Mirek, my intention why I want it as member functions is, because my RPC Methods all need a database connection, so in one class function I open the connection and keep it open. And because I don't want to pass this PostgreSqlSession to all RPC Functions, I want to use a member. As Mirek already said, global instance is just fine for this purpose. If you don't like global variables, than you can use the template Single<T>, just as I did in the example, to create single instance of some type T, which is shared by all the code that calls Single<T>(). Furthermore, for Sql connection, there are already application-wide variables SQL and SQLR for this (the second one can be used for readonly connections). These variables even allow to write some sql commands in simpler way.

NilaT wrote on Sun, 05 February 2017 11:02Furthermore I need an Instance of a very big class and don't want to pass this either, so a member seems to be the best solution in my opinion.No U++ solution for this, but again, global variable or any kind of singleton should do.

NilaT wrote on Sun, 05 February 2017 11:02PS: Maybe you can explain this code? I mean... it compiles, but I really don't know what this does:
[](RpcData& rpc){Single<MyRpcClass>().Status(rpc);});Single<> creates singleton instance, while "[](RpcData& rpc) {}" is a definition of lambda function from C++11. It is basically an anonymous function that takes RpcData& as parameter and calls your member function on the singleton as mentioned above.

Honza